Software Transactional Networking: Concurrent and Consistent Policy Composition

Dan Levin
Marco Canini, Petr Kuznetsov*, Stefan Schmid

TU Berlin/T-Labs, *Telecom ParisTech
Software Transactional Networking: Concurrent and Consistent Policy Composition
Network Policy Specification
Network-wide Policy: Not Monolithic

Policy may originate from multiple authors, defined across multiple functional modules.

...necessitates policy composition prior to network update.
1. Precedence must be defined across policy sources
2. Packet forwarding rule priorities must be defined, and respect policy source precedence
Now, consider policy composition in the distributed control plane...

Controller Application
Routing     Monitoring     Waypoint

Network Information Base

Switch
Reader-Writer Model

Controller Replication Model

Control Application Factorization
Research Question #1

How does the design of the **distributed control plane** affect policy composition with respect to:

1. The consistency of the composition
2. The semantics of consistent network update
Distributed Control Models

1: Hot Standby Replication

Master

- Routing
- Monitoring
- Waypoint

Slave

- Routing
- Monitoring
- Waypoint

Composition

1-Writer
Distributed Control Models
2: Sharding by Disjoint Flow-Space
Distributed Control Models
3: Sharding by Policy

Controller Application

Routing  Monitoring

n-Writer

Controller Application

Waypoint
Concurrency Gone Wrong

Impossible to guarantee a deterministic outcome without policy synchronization
Deterministic Policy Composition which respects precedence of multi-authorship
Research Question #2

Can we realize a distributed policy composition interface that...

<table>
<thead>
<tr>
<th>is agnostic to:</th>
<th>guarantees:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Distribution model</td>
<td>Consistent update semantics</td>
</tr>
<tr>
<td>Switch reader-writer model</td>
<td>Consistent policy composition</td>
</tr>
</tbody>
</table>
Software Transactional Memory

process 1
read
write

process 2
read
write

process 3
read
write

Transactional Interface

Shared Datastructure
Software Transactional Networking

Routing  Monitoring  Waypoint

read  write  read  write

Transactional Interface
The STN Interface

Routing

Monitoring

Waypoint

Apply(p) → Ack

Apply(p) → Nack(reason)

STN Interface
Conceptualizing STN

Controller Platform

Controller Application

Routing

Monitoring

Waypoint

Apply(p)

Ack

Apply(p)

Nack(reason)

STN Middleware

Controller Platform

Lock

Lock

Lock

Lock

Lock

Lock

Atomic Read-Modify Write
Begin consistent network updates

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>src=10.1.0/24</td>
<td>fwd(IPS)</td>
</tr>
<tr>
<td>tcp=80</td>
<td>count + fwd</td>
</tr>
</tbody>
</table>
STN in Action (Nack Case)

Routing

Waypoint

Apply(p1) → Ack

Apply(p2) → Nack(reason)

STN Interface

Begin consistent network updates

Match | Action
--- | ---
src=10.1.0/24 | fwd(IDS)
dst=10.1/16 | fwd(2)
STN fine-grained locking algorithm

1 apply(*policy p, policy tree f*):
2     tag = newTag(p)
3     for s in switches():
4         if s.rules.doCompose(p, f):
5             tag’ = concurrentTag(s, tag)
6             s.addTagRules(p, tag’)
7         else remove tag’ rules, nack(reason)
8     for s in ingressSwitches():
9         s.addRule(match=*, action=push(tag’))
Open Issues

- For our simple algorithm, tag and forwarding-rule state grows exponentially $O(2^n)$ for $n$-concurrency
- Different Update Consistency and Policy Composition Semantics
- Atomic read-modify-write primitive at switch
- Controller fault tolerance
Summary

Full technical report for more details
http://arxiv.org/abs/1305.7429

Internship Opportunities @T-Labs

- Interface for distributed policy composition
- Framework to reason about concurrent policy composition and consistent update
Backup Slides
Policy Composition Review

<table>
<thead>
<tr>
<th>Priority</th>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dst=10.1/16, *</td>
<td>fwd(1)</td>
</tr>
<tr>
<td>1</td>
<td>dst=10.2/16, *</td>
<td>fwd(2)</td>
</tr>
<tr>
<td>2</td>
<td>dst=10.1/16,dport=80, *</td>
<td>fwd(1)</td>
</tr>
<tr>
<td>2</td>
<td>dst=10.2/16,dport=80, *</td>
<td>fwd(2)</td>
</tr>
<tr>
<td>3</td>
<td>dst=10.1/16, src=10.1.0/24, dport=80, *</td>
<td>fwd(IDS)</td>
</tr>
</tbody>
</table>

Routing:  dst=10.1/16 → fwd(1)  
dst=10.2/16 → fwd(2)

Monitor:  tcp_port=80 → count

Waypoint:  src=10.1.0/24 → fwd(IDS)

Composition is the “cross-product” of rules.
Distributed Control Models
2: Sharding by Region